BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

(12) UK Patent Application (19) GB (11) 2 227 339 (13) A

(43) Date of A publication 25.07.1990

(21) Application No 8901349.4

(22) Date of filing 21.01.1989

(51) INT CL[5]
G06F 15/40

(52) UK CL (Edition K)
G4A AUD

(56) Documents cited
None

(58) Field of search
UK CL (Edition J) G4A APX AUD

(71) Applicant
Kashinath Narayan Dandeker
c/o School Of Info. & Comp. Studies, HCHE,
Cottingham Road, Hull, HU6 7RT, United Kingdom

(72) Inventor
Kashinath Narayan Dandeker

(74) Agent and/or Address for Service
Kashinath Narayan Daneker
c/o School of Info. & Comp. Studies, HCHE,
Cottingham Road, Hull, HU6 7RT, United Kingdom

(54) Complex expression generator

(57) This invention relates to the generation of complex expressions from simple expressions defined as macros, where each expression represents a structure which is equivalent to a specified procedure that has to be carried out in order to obtain ultimately the desired result. The required macro processing can be carried out manually but is speeded up if a suitable macro processor is used. The generated complex expression after macro processing is resolved in one step without re-arrangement by a suitable compiler. An application is implementation of parameterized views in Relational Query Languages, exploiting the link between the BNF Grammar of the Relational Query Language and the replacement texts of the macro definitions.

GB 2 227 339 A

2227339

COMPLEX EXPRESSION GENERATOR

This invention relates to the generation of complex expressions from simple expressions defined as macros, where each expression represents a structure which is equivalent to a specified procedure that has to be carried out in order to obtain ultimately the desired result. The required macro processing can be carried out manually but is speeded up if a suitable macro processor is used. The generated complex expression after macro processing is resolved in one step without re-arrangement by a suitable compiler.

In Relational Query Languages a view facility is provided in order to be able to resolve efficiently a complicated query in one step by defining it in terms of simpler queries where each simple query represents a step that otherwise would have to be carried out in order to obtain ultimately the desired information. A view has a name and an associated query definition which represents the parse tree, a structure which is equivalent to a specified procedure that has to be carried out in order to obtain the required information as specified by the associated query.

At some stage in the processing of a view the name of the view is replaced by the wording of its associated query. As this process appears to be similar to macro processing it was thought possible to implement views in Relational Query Languages by defining views as macros where the replacement text of the macro definition contains the query definition associated with the view. The complex query generated after macro processing might be re-arranged before it was resolved

in one step by the Relational Query language compiler that was used. This invention has enabled that possibility to be realized. Views can be defined as macros and these can be implemented in Relational Query Languages with a suitable macro processor. Moreover the generated complex query after macro processing can be resolved in one step without re-arrangement by the Relational Query Language compiler that was used.

If views are defined as macros it becomes possible to define a parameterized view as a parameterized macro in which the replacement text contains the same parameterized query definition associated with the parameterized view, so that all the information required to generate the specified procedure that has to be carried out to obtain the required information as specified by the parameterized query, is not available at the time of the definition of such a parameterized view. A parameterized view is a new concept which has not been implemented with the use of the existing methods for implementing views in Relational Query Languages. However if the parameterized view is referenced subsequently in a query and the appropriate parameters are provided at that time and processed then all the information required to generate the specified procedure that has to be carried out becomes available and the desired result can be obtained. Parameterized views are implemented in Relational Query languages with a macro processor which can substitute text, replace parameters and has the ability to handle nested macro calls.

It was discovered that this method of implementing views as macros in Relational Query Languages depended solely on the link between the BNF Grammar of the Relational Query Language that was used and the replacement text of the macro definitions used. It was realized that this property was not limited to Relational Query Languages and query expressions but could be extended to any other type of language and to any other types of expressions or sets of symbols or structures which could be defined as macros so that the generated complicated expression or set of symbols or structure after macro processing could be resolved in one step without re-arrangement by the language compiler that was used.

This invention exploits the link between on the BNF Grammar of the language and the replacement text of the macro definitions used.

The BNF Grammar of the language should be as follows:

Sentence Symbol of the language := Identifier
| Any Expression
| Any Set of Symbols
| Any Structure
| (Any Expression)
| (Any Set of Symbols)
| (Any Structure) .

where Any Expression is a query expression, an arithmetic expression or any other expression and where "(" and ")" are a specified pair of delimiters and where complex expressions or complex sets of symbols or complex structures are made

from simple expressions or simple set of symbols or simple structures by using connecting operators of the types shown in the following examples :

Any Expression := Any Expression connecting_operator Any Expression

where Any Expression is a query expression, or an arithmetic expression or any other expression and where connecting_operator is a suitable operator.

In this case the macro definition is of the form where the macro name is an Identifier and the replacement text of which is (Any Expression) where "(" and ")" are the same pair of delimiters specified in the BNF Grammar of the language used.

Any Set of Symbols := Any Set of Symbols connecting_operator Any Set of Symbols

where Any Set of Symbols is a set of any symbols and where connecting_operator is a suitable operator.

In this case the macro definition is of the form where the macro name is an Identifier and the replacement text of which is (Any Set of Symbols) where "(" and ")" are the same pair of delimiters specified in the BNF Grammar of the language used.

Any Structure := Any Structure connecting_operator Any Structure

where Any Structure is any structure and where connecting_operator is a suitable operator.

In this case the macro definition is of the form where the macro name is an Identifier and the replacement text of

which is (Structure) where "(" and ")" are the same pair of delimiters as specified in the BNF Grammar of the language used.

The examples that illustrate the use of this invention will be limited to the implementation of parameterized views in Relational Query Languages with suitable macro processors. These examples will show different Relational Query Languages being used with different macro processors.

If we use the pre-processor to the C compiler provided with the Unix System as the macro processor to implement views in the Dundee form of the DEAL language then the two parameterized view definitions for citysup and impsup in the form of two parameterized macro definitions named citysup with formal parameter x and impsup with formal parameters x and y in their replacement texts and the corresponding macro call with the two actual parameters london and 30 which are provided would be as follows:

```
#define citysup(x) ( supplier where city = 'x' )
#define impsup(x,y) ( citysup(x) where status > y )
impsup(london,30)
```

on processing the above macro definitions the following complex query is generated which is resolved without rearrangement by the compiler.

```
( ( supplier where city = 'london' ) where status > 30 )
```

The above query is equivalent to the following single query:

```
supplier where city = 'london' and status > 30
```

If we use the m4 macro processor provided with the Unix System to implement views in the SQL form of the DEAL

language then the two parameterized view definitions for citysup and impsup in the form of two parameterized macro definitions named citysup with positional parameter $1 and impsup with positional parameters $1 and $2 in their replacement texts and the corresponding macro call with the two actual parameters london and 30 which are provided would be as follows:

```
define(citysup,( select * from supplier where city = '$1' ))
define(impsup,( select * from citysup($1) where status > $2
))
impsup(london,30);
```

on processing the above macro definitions the following complex query is generated which is resolved without rearrangement by the compiler.

```
( select * from ( select * from supplier where city ='lon-
don' ) where status > 30 );
```

The above query is equivalent to the following single query:

```
select * from supplier where city = 'london' and status >
30;
```

If we use a custom built pre-processor to implement views in the Dundee form of the DEAL language then the two parameterized view definitions for citysup and impsup in the form of two parameterized macro definitions named citysup with formal parameter x and impsup with formal parameters x and y in their replacement texts and the corresponding macro call with the two actual parameters london and 30 which are provided would be as follows:

```
citysup(x) == ( anon[all] from supplier where city ='x' )
impsup(x,y) == ( anon[all] from citysup(x) where status > y
```

```
)
impsup(london,30);
```

on processing the above macro definitions the following complex query is generated which is resolved without rearrangement by the compiler

```
( anon[all] from ( anon[all] from supplier where city =
'london' ) where status > 30 );
```

The above query is equivalent to the following single query:

```
anon[all] from supplier where city = 'london' and status >
30;
```

CLAIMS

1. A COMPLEX EXPRESSION GENERATOR enables parameterized views to be implemented in Relational Query Languages. The required macro processing can be carried out manually but it is speeded up with the use of a macro processor that can substitute text, replace parameters and can handle nested macro calls.

2. Parameterized views cannot be implemented in Relational Query Languages using existing methods followed in current database practice.

3. Use of a COMPLEX EXPRESSION GENERATOR is a cheaper and simpler method of implementing views in Relational Query Languages than using existing methods followed in current database practice.

4. As the complex expression generated after macro processing by the COMPLEX EXPRESSION GENERATOR is in the Relational Query language used, it is possible to read it, to decipher it and to ensure that it meets the user requirements before using it to generate the specified procedure that has to be carried out to obtain eventually the desired result. Such a procedure cannot be carried out using existing methods followed in current database practice.

5. Use of a COMPLEX EXPRESSION GENERATOR is not limited to Query expressions and Relational Query Languages. It can be used without change with Arithmetic expressions or any other expressions or any sets

of symbols or any structures provided a suitable compiler is available.' The complex expression generated after macro processing is resolved in one step withot re-arrangement by a suitable compiler.

6. The COMPLEX EXPRESSION GENERATOR is designed to exploit the link between the BNF Grammar of the language and the replacement text of the macro definitions used and is based on theoretical foundations which have been researched thoroughly.

Amendments to the claims have been filed as follows

1. The COMPLEX EXPRESSION GENERATOR comprising macro definitions, which may be parameterised, followed by their corresponding macro calls, which may include parameter values that enables complex sentences to be generated in a language from simple sentences in that language by a subsequent process of macro expansion which excludes syntax checking, parsing and code generation.

2. The macro definitions, macro calls and the associated parameter values that comprise the COMPLEX EXPRESSION GENERATOR as claimed in Claim 1 are entered and are not generated as a result of any process.

3. The complex sentences generated by the COMPLEX EXPRESSION GENERATOR as claimed in Claim 1 can be read, checked and understood easily before these are used to perform the equivalent specified procedures so as to obtain eventually the desired result. Such a procedure cannot be carried out by a process which excludes syntax checking, parsing and code generation using existing methods followed in current practice.

4. The complex sentences generated as claimed in Claim 3 by The COMPLEX EXPRESSION GENERATOR as claimed in Claims 1 include @uery expressions, Arithmetic expressions or any other expressions or any sets of symbols or any structures because the macro definitions, macro calls and the associated parameter values that comprise

the COMPLEX EXPRESSION GENERATOR as claimed in Claim 1 are entered as claimed in Claim 2 in an appropriate form and are subjected to an appropriate subsequent process of macro expansion which excludes syntax checking, parsing or code generation.

5. The complex sentences generated as claimed in Claim 3 or Claim 4 by the COMPLEX EXPRESSION GENERATOR as claimed in Claim 1 after a subsequent process of macro expansion, which excludes syntax checking, parsing or code generation, are resolved in one step without rearrangement by a suitable compiler.

6. The complex sentences generated as claimed in Claim 3 or Claim 4 by the COMPLEX EXPRESSION GENERATOR as claimed in Claim 1 after a subsequent process of macro expansion, which excludes syntax checking, parsing or code generation are due to the link between the BNF Grammar of the language and the macro definitions, the macro cells and the parameter values provided as claimed in Claim 2.